Course contents (plan)

1. Course basics, intro
2. Sw Eng in general, overview
3. **Requirements**
4. Different software systems
5. Basic UML Diagrams (”Class”, Use Case, Navigation)
6. Life Cycle models
7. UML diagrams, in more detail
8. Quality and Testing
9. Project work
10. Project management
11. Open source, APIs, IPR
12. Embedded systems
13. Recap

3. Requirements (SRS = sw req. specification)

- Preliminary Study / Feasibility Analysis
- requirements elicitation
- stakeholders and stakeholder analysis
- Requirements
  - functional
  - non-functional
  - restrictions
- safety vs. security
- usability
- producer’s/developer’s responsibilities
- customer’s responsibilities
- requirements management (e.g. traceability)

Current at course

- we have nine project groups

- **WE3 groups this week are on WED and THU**
- after those we think next week WE4 groups

- ***A shorter lecture session today 17.09.2019.***

SO I'M READY TO START CODING. DO YOU HAVE THE SPEC?
NOT EXACTLY.
HOW ABOUT AN OUTLINE? OR SOME SCREEN SHOTS?
NO TIME.
WHAT DO YOU HAVE, THEN?
A SCENARIO DIORAMA.
Copyright 2006 Hans Bjordahl
TELL ME YOU'RE KIDDING.
JUDGING BY THE DIORAMA, I THINK WE SHOULD USE MANAGED CODE.

Bug Bash by Hans Bjordahl http://www.bugbash.net/

Feasibility Study, Preliminary Analysis

- usually made by the vendor/developer (sw company)
- is made at the very beginning, after the idea of a new system
- a few hours … a few working days is spent
- is made to find out whether a project will be started, or not
- do we have enough time to do such a project, skilled workers, tools, money, possible other/further customers in sight,…
- in many cases the result is that such a project is not started (perhaps later e.g. when technology will be more mature)
- if project is started, matters of Feasibility Study will be described in more detail in Requirements Specification and Project Plan.
- **BTW: the first thing is to check internet if such thing has already been done ! ”Do not re-invent the wheel.”**

Start for requirements, one path

- customer has some ideas and thoughts about requirements
- developer company writes first basic requirements (based on customer’s data)
- developer makes stakeholder search and analysis (which to include)
- developer (re-)defines basic requirements, GUI sketch
- requirements are discussed between customer and developer
- developer writes more concrete requirements (more details), GUI proto
- end-users are taken into account, GUI proto 2
- requirements and GUI are defined in detail
- requirements are discussed in detail between customer and developer
- requirements specification document (includes GUI)

…requirements alternate/interchange/vary at least slightly during the project…

[**Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010**]

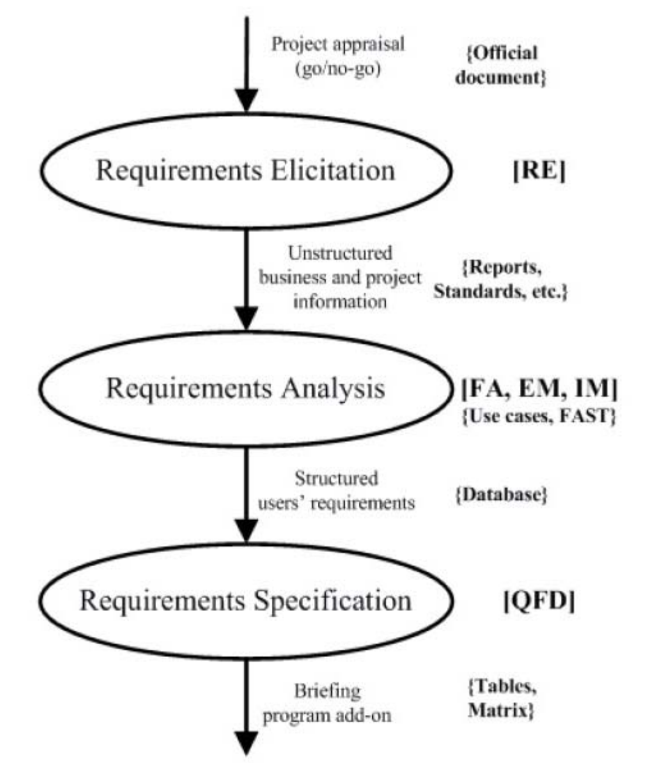

Project appraisal (go/no-go)
{Official document}
Requirements Elicitation
[RE]
Unstructured business and project information
{Reports, Standards, etc.}
Requirements Analysis
[FA, EM, IM]
{Use cases, FAST}
Structured users' requirements
{Database}
Requirements Specification
[QFD]
Briefing program add-on
{Tables, Matrix}

Fig. 2. Stages of the methodology.

RE = Requirements Engineering

FA = Functional Analysis

IM = Information Modelling

FAST = Functional Analysis System Techniques

QFD = Quality Function Deployment

[**Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010**]

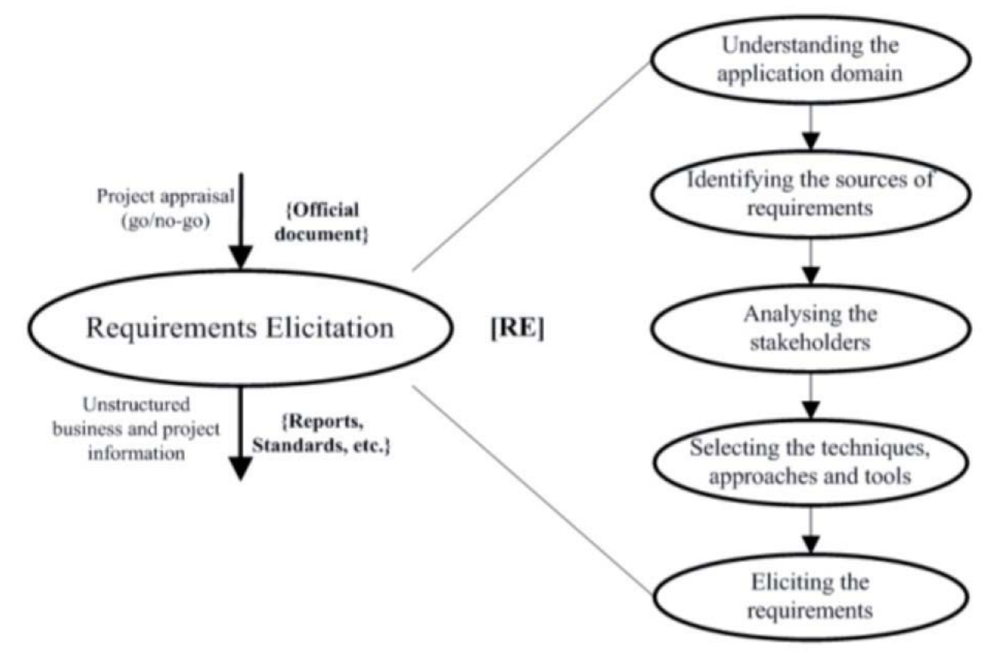

Project appraisal (go/no-go)
{Official document}
Requirements Elicitation
[RE]
Unstructured business and project information
{Reports, Standards, etc.}
Understanding the application domain
Identifying the sources of requirements
Analysing the stakeholders
Selecting the techniques, approaches and tools
Eliciting the requirements

Fig. 3. Stage 1: the requirements elicitation process in detail.

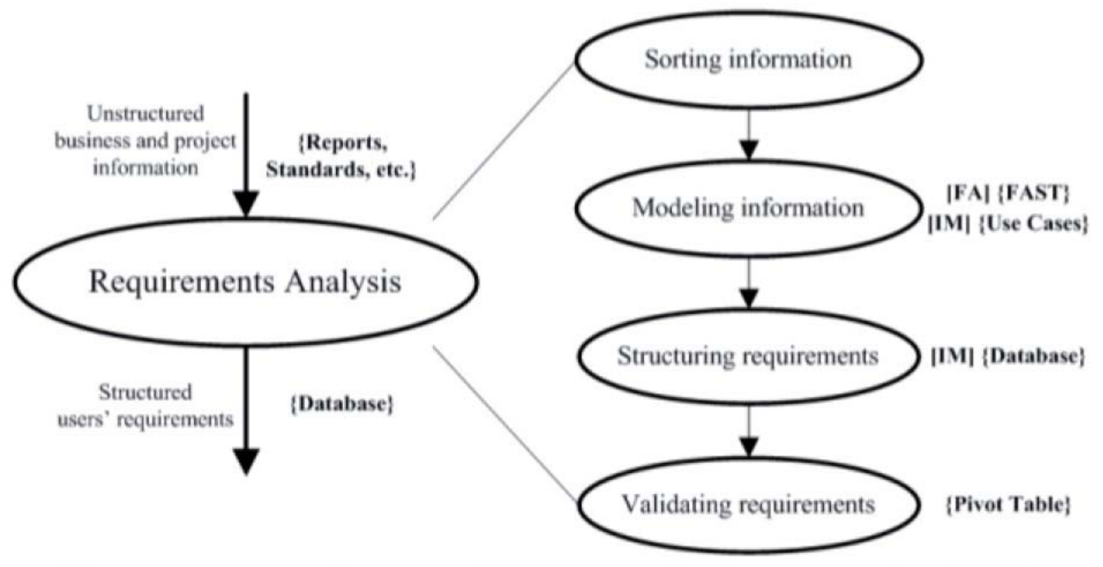


Fig. 4. Stage 2: the requirements analysis process in detail.

[**Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010**]

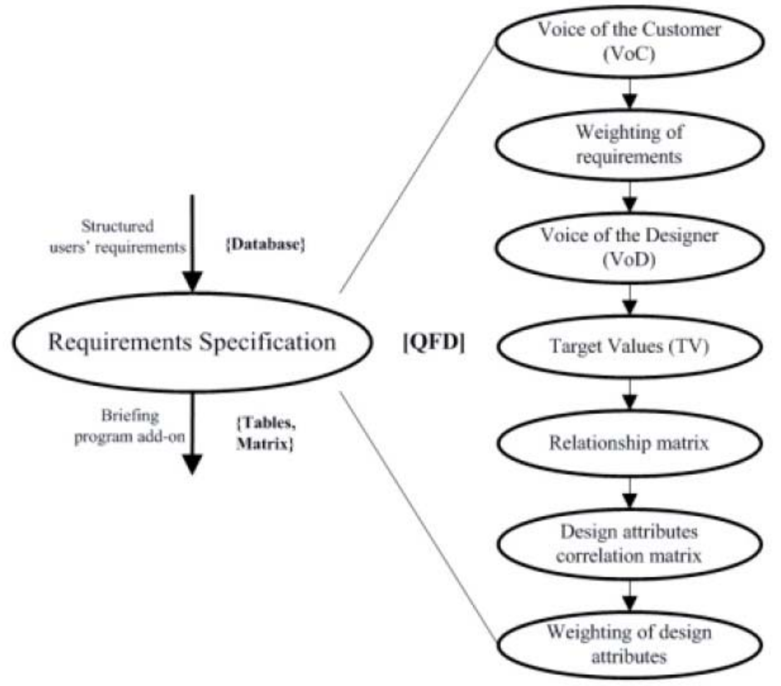

Structured users' requirements
{Database}
Requirements Specification
[QFD]
Briefing program add-on
{Tables, Matrix}
Voice of the Customer (VoC)
Weighting of requirements
Voice of the Designer (VoD)
Target Values (TV)
Relationship matrix
Design attributes correlation matrix
Weighting of design attributes

Fig. 6. Stage 3: the requirements specification process in detail.

QFD = Quality Function Deployment

[**Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010**]

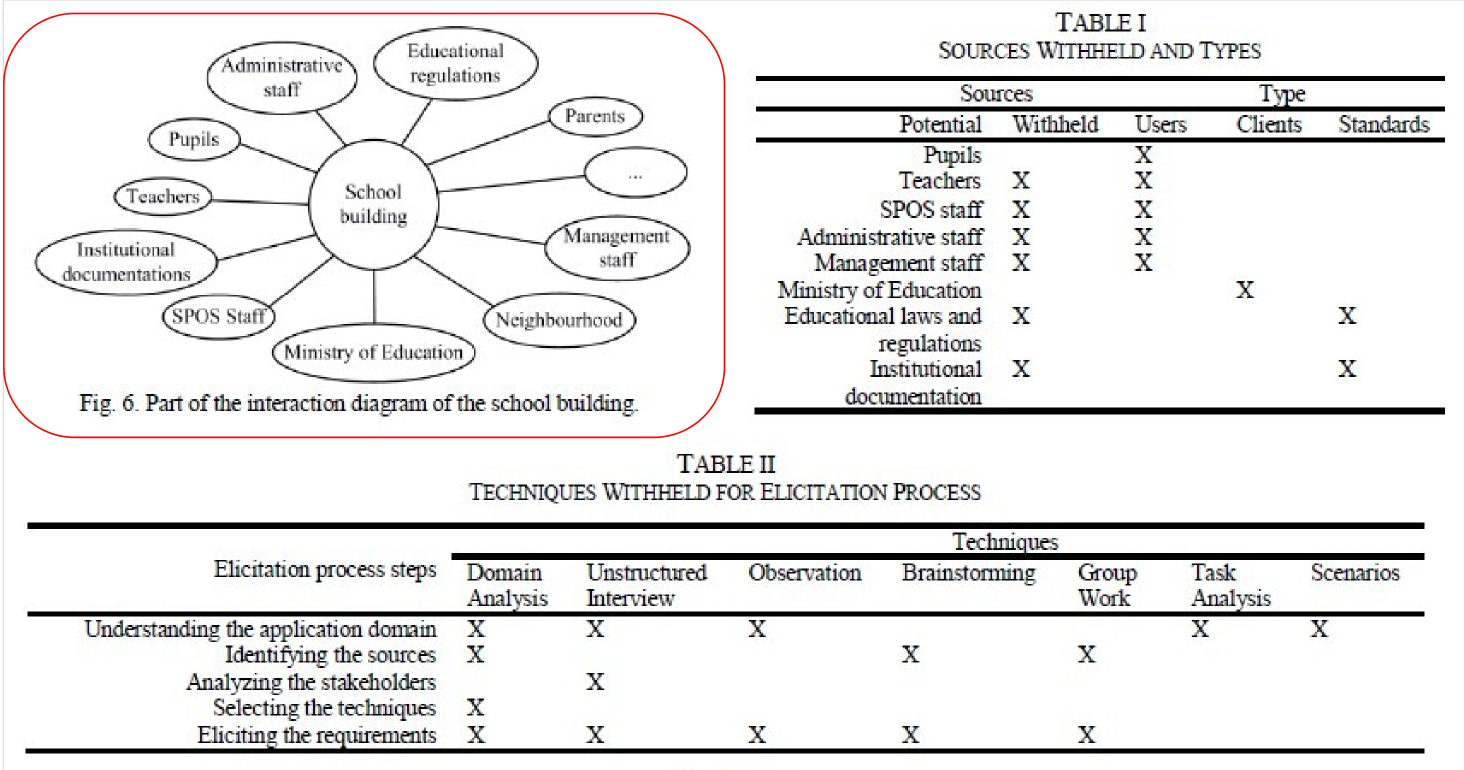


Fig. 6. Part of the interaction diagram of the school building.

TABLE I
SOURCES WITHHELD AND TYPES

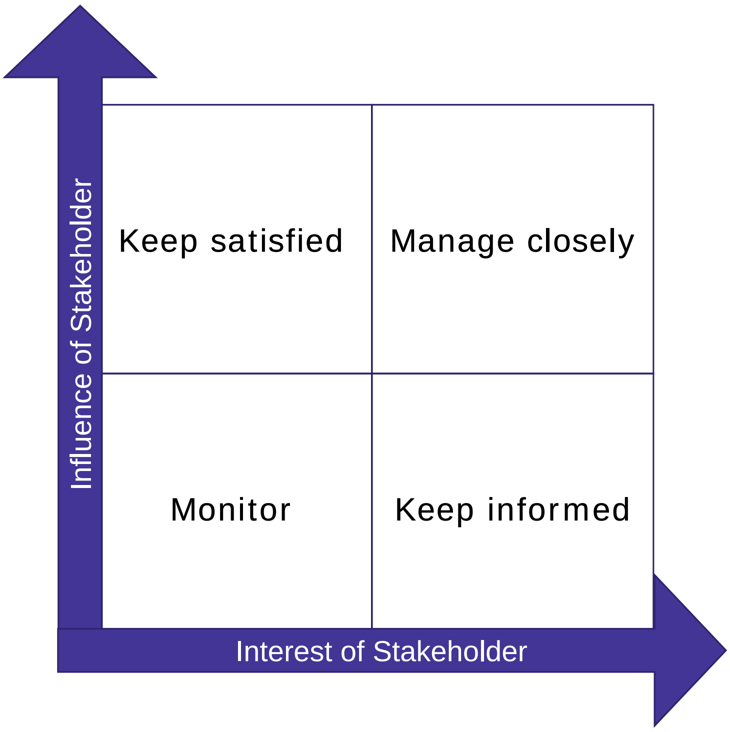
Sources		Type		
Potential	Withheld	Users	Clients	Standards
Pupils		X		
Teachers	X	X		
SPOS staff	X	X		
Administrative staff	X	X		
Management staff	X	X		
Ministry of Education			X	
Educational laws and regulations	X			X
Institutional documentation	X			X

TABLE II
TECHNIQUES WITHHELD FOR ELICITATION PROCESS

Elicitation process steps	Techniques						
	Domain Analysis	Unstructured Interview	Observation	Brainstorming	Group Work	Task Analysis	Scenarios
Understanding the application domain	X	X	X			X	X
Identifying the sources	X			X	X		
Analyzing the stakeholders		X					
Selecting the techniques	X						
Eliciting the requirements	X	X	X	X	X		

[Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: the elicitation process, 2010]

Stakeholders



Keep satisfied

Manage closely

Monitor

Keep informed

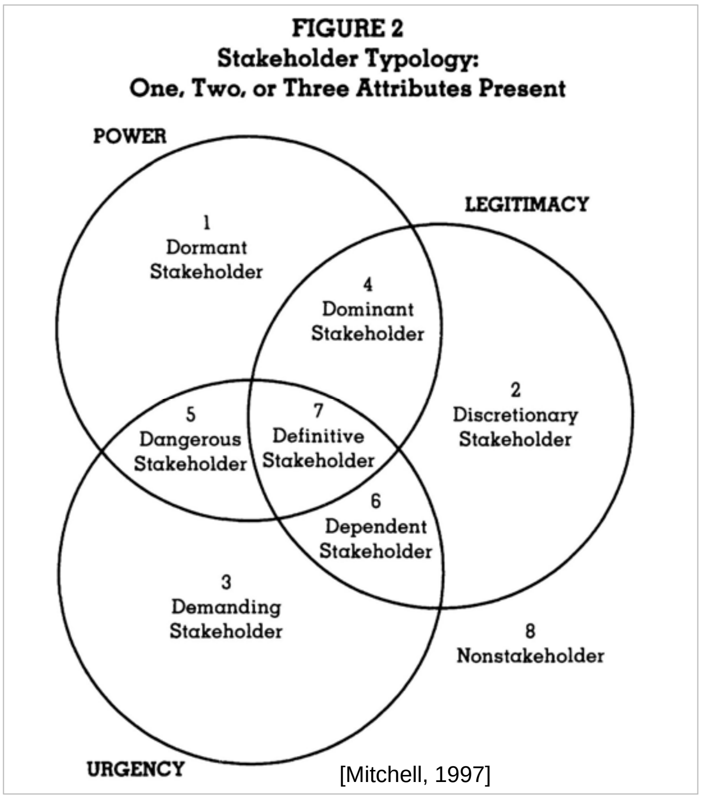
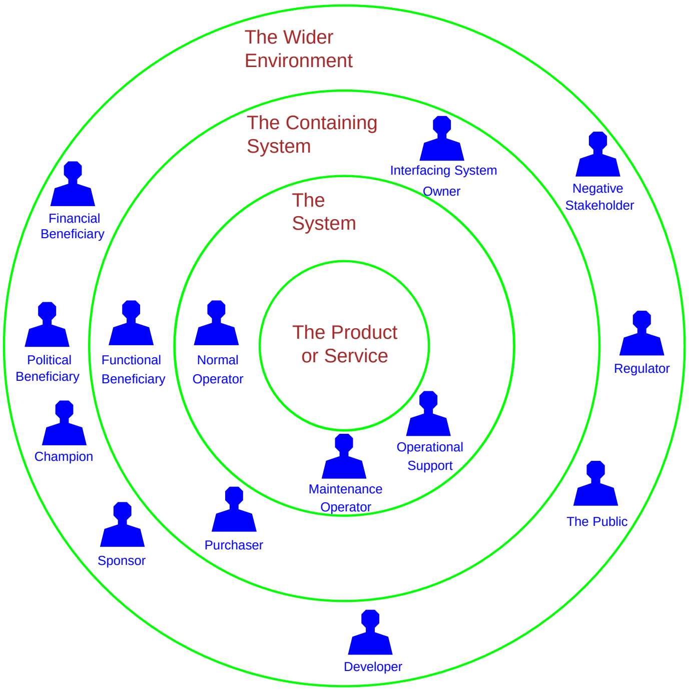


FIGURE 2
Stakeholder Typology:
One, Two, or Three Attributes Present

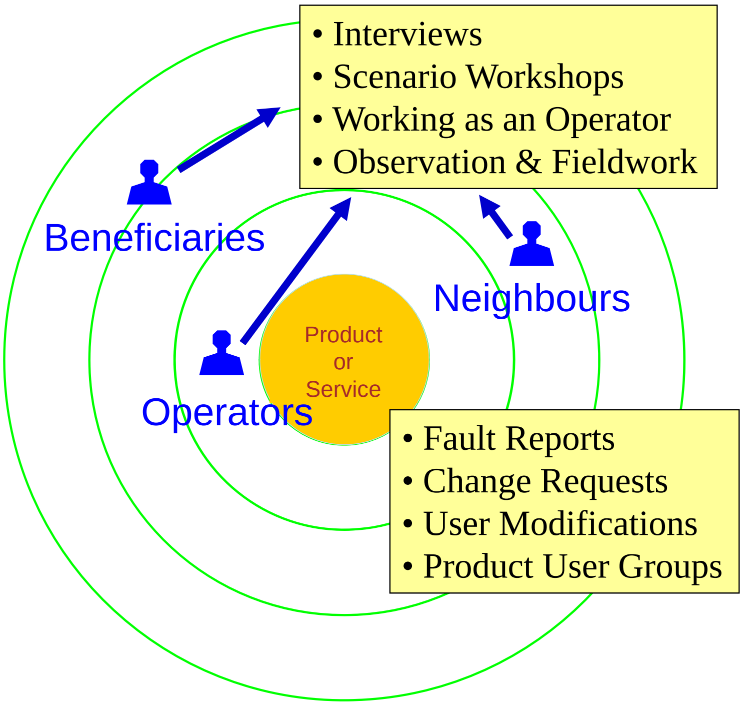
[Mitchell, 1997]

Stakeholders in a Typical System



[scenarioplus.org.uk]

Discovery Contexts and Sources

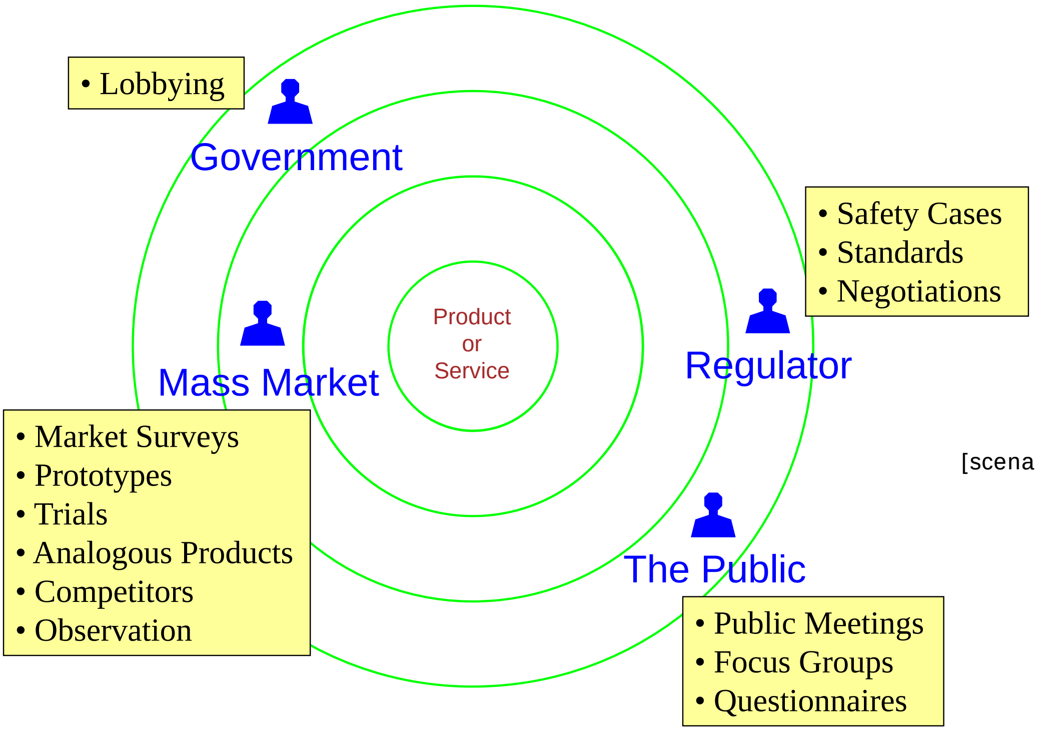


- Interviews
- Scenario Workshops
- Working as an Operator
- Observation & Fieldwork

- Fault Reports
- Change Requests
- User Modifications
- Product User Groups

[scenarioplus.org.uk]

Discovering from Non-Operational Roles



- Lobbying

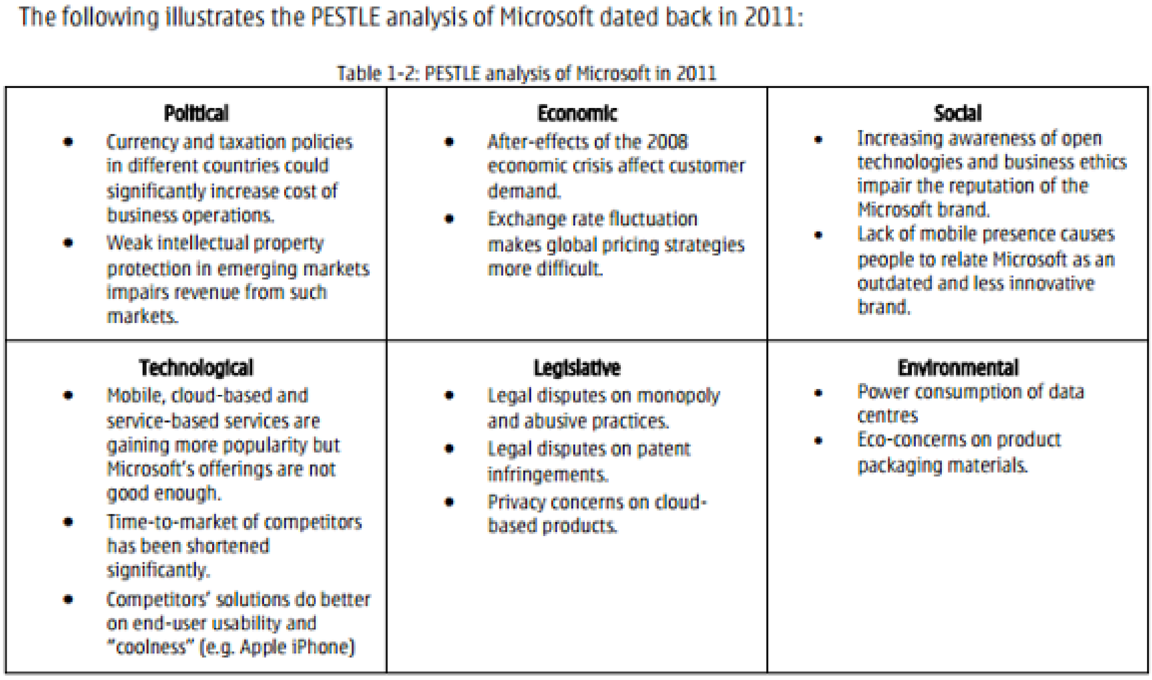
- Safety Cases
- Standards
- Negotiations

- Market Surveys
- Prototypes
- Trials
- Analogous Products
- Competitors
- Observation

- Public Meetings
- Focus Groups
- Questionnaires

[scenarioplus.org.uk]

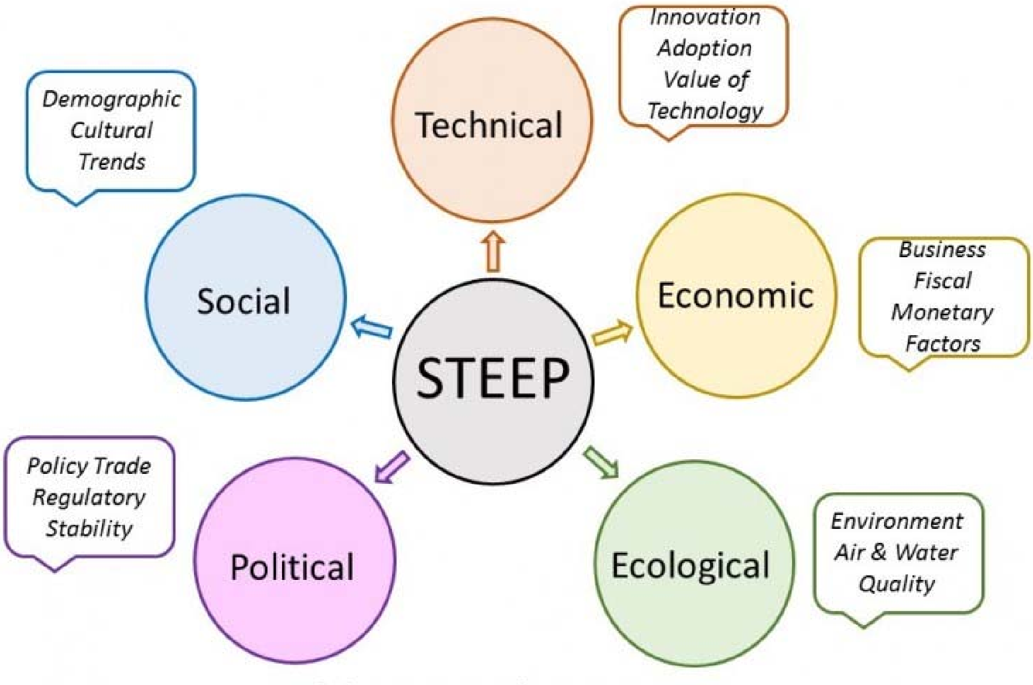
PESTLE… STEEPLE… stakeholders



The following illustrates the PESTLE analysis of Microsoft dated back in 2011:

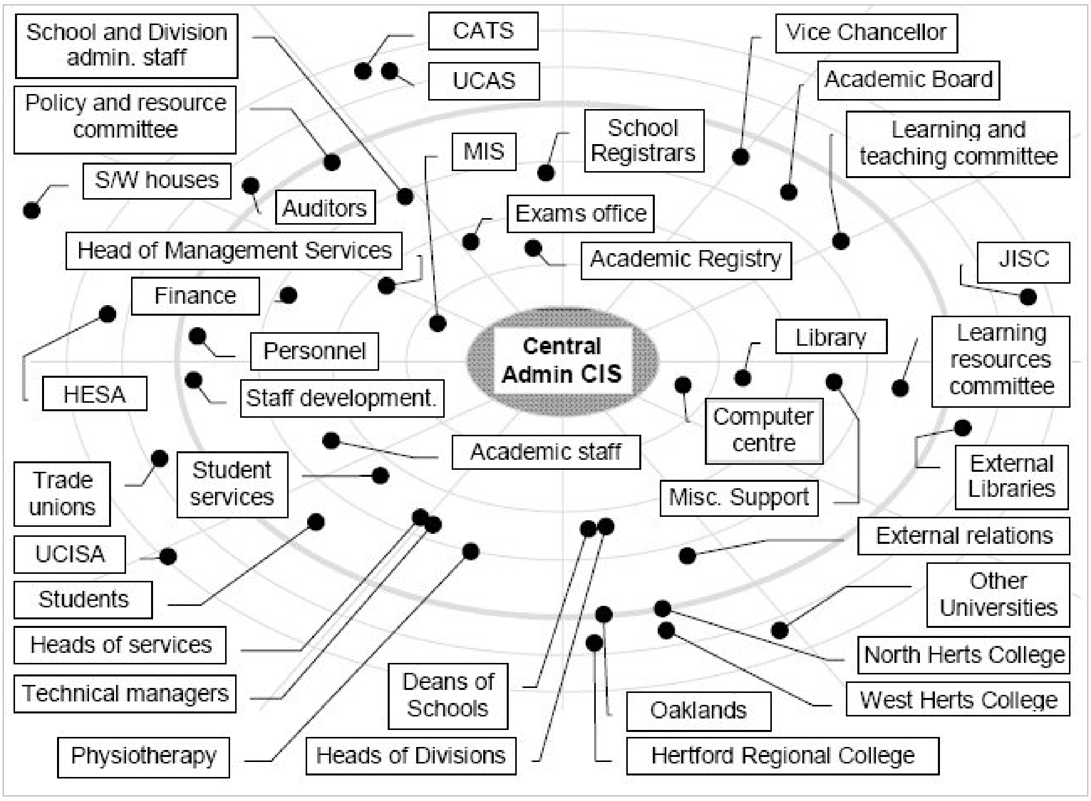
Table 1-2: PESTLE analysis of Microsoft in 2011

Political	**Economic**	**Social**
• Currency and taxation policies in different countries could significantly increase cost of business operations. • Weak intellectual property protection in emerging markets impairs revenue from such markets.	• After-effects of the 2008 economic crisis affect customer demand. • Exchange rate fluctuation makes global pricing strategies more difficult.	• Increasing awareness of open technologies and business ethics impair the reputation of the Microsoft brand. • Lack of mobile presence causes people to relate Microsoft as an outdated and less innovative brand.
Technological	**Legislative**	**Environmental**
• Mobile, cloud-based and service-based services are gaining more popularity but Microsoft's offerings are not good enough. • Time-to-market of competitors has been shortened significantly. • Competitors' solutions do better on end-user usability and "coolness" (e.g. Apple iPhone)	• Legal disputes on monopoly and abusive practices. • Legal disputes on patent infringements. • Privacy concerns on cloud-based products.	• Power consumption of data centres • Eco-concerns on product packaging materials.


Demographic Cultural Trends
Technical
Innovation Adoption Value of Technology
Social
STEEP
Economic
Business Fiscal Monetary Factors
Policy Trade Regulatory Stability
Political
Ecological
Environment Air & Water Quality

©The Business Intelligence Source

University example



School and Division admin. staff

CATS

UCAS

Vice Chancellor

Academic Board

Policy and resource committee

MIS

School Registrars

Learning and teaching committee

S/W houses

Auditors

Exams office

Head of Management Services

Academic Registry

JISC

Finance

Personnel

Central Admin CIS

Library

Learning resources committee

HESA

Staff development.

Computer centre

Academic staff

External Libraries

Trade unions

Student services

Misc. Support

UCISA

External relations

Students

Other Universities

Heads of services

North Herts College

Technical managers

Deans of Schools

West Herts College

Oaklands

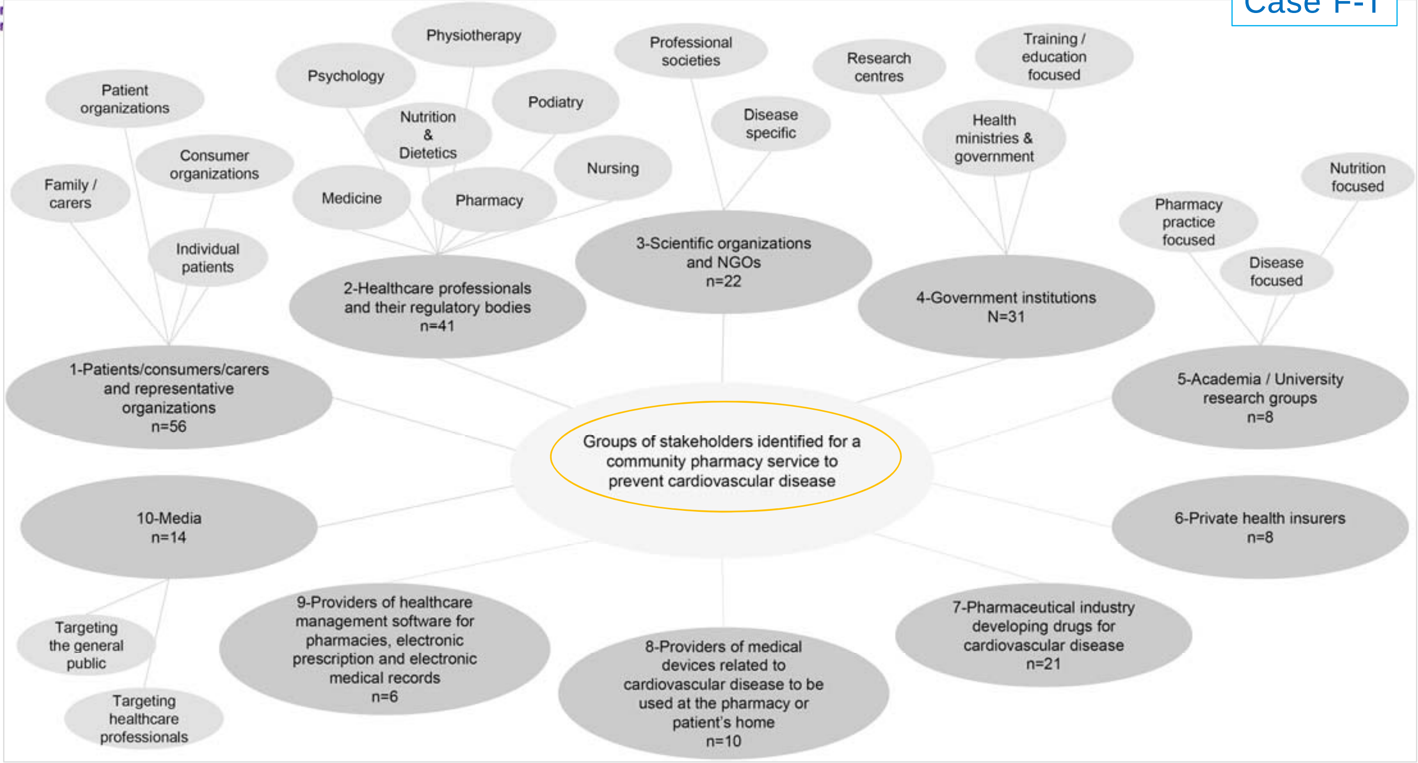
Physiotherapy

Heads of Divisions

Hertford Regional College

[scenarioplus.org.uk]

Case F-T



Groups of stakeholders identified for a community pharmacy service to prevent cardiovascular disease

1-Patients/consumers/carers and representative organizations n=56

Patient organizations

Consumer organizations

Family / carers

Individual patients

2-Healthcare professionals and their regulatory bodies n=41

Physiotherapy

Psychology

Podiatry

Nutrition & Dietetics

Nursing

Medicine

Pharmacy

3-Scientific organizations and NGOs n=22

Professional societies

Disease specific

4-Government institutions N=31

Research centres

Training / education focused

Health ministries & government

5-Academia / University research groups n=8

Nutrition focused

Pharmacy practice focused

Disease focused

6-Private health insurers n=8

7-Pharmaceutical industry developing drugs for cardiovascular disease n=21

8-Providers of medical devices related to cardiovascular disease to be used at the pharmacy or patient's home n=10

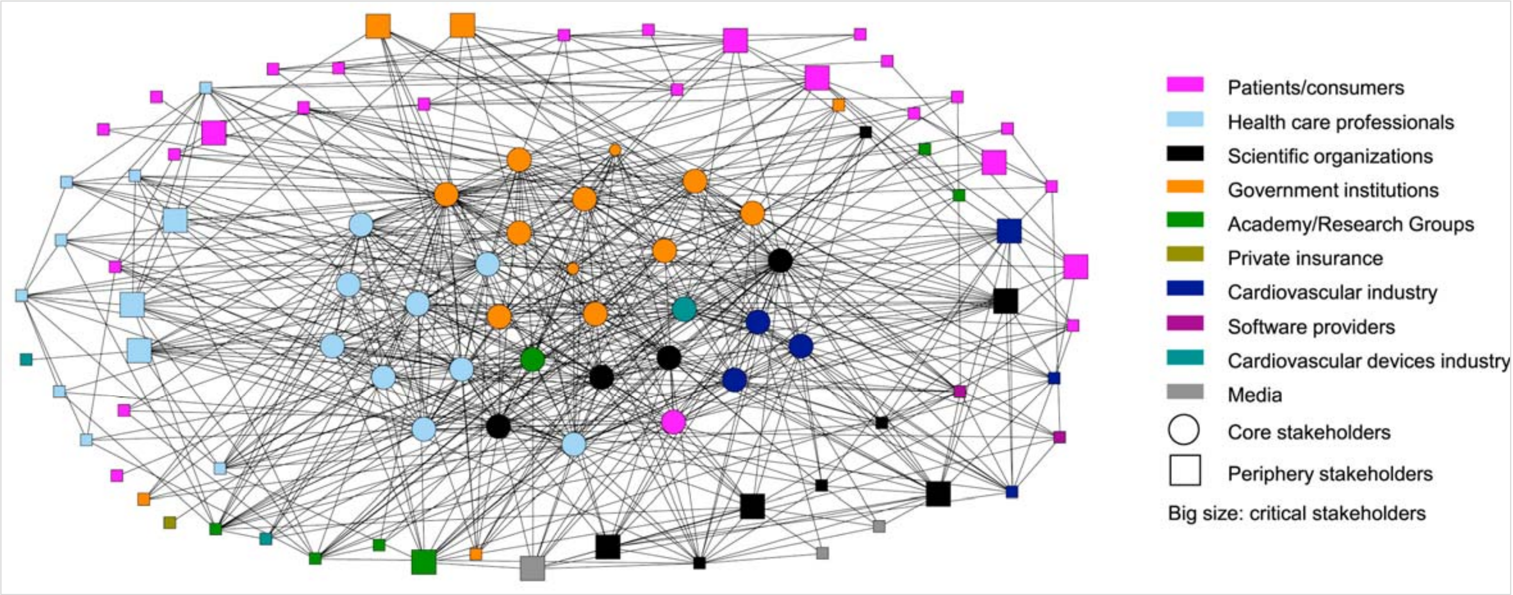
9-Providers of healthcare management software for pharmacies, electronic prescription and electronic medical records n=6

10-Media n=14

Targeting the general public

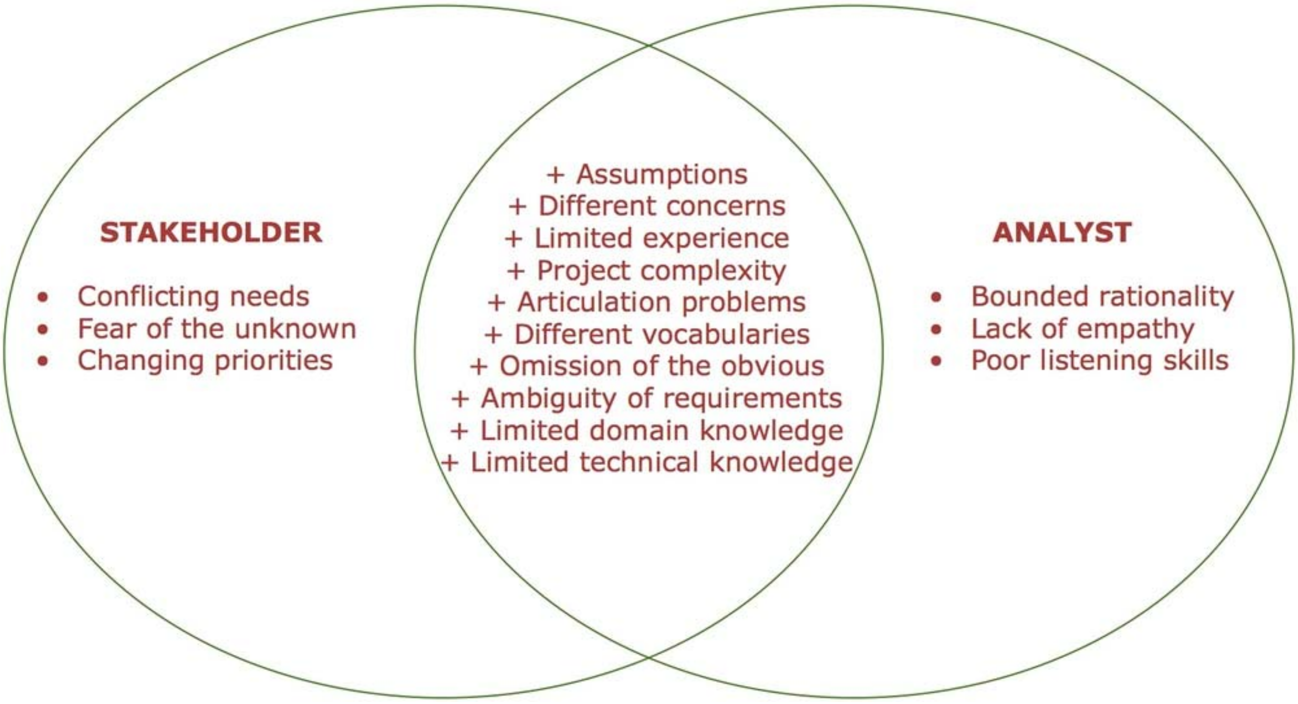
Targeting healthcare professionals

Case F-T: Collaboration network about stakeholders.



[L. Franco-Trigo et al.: Collaborative health service planning: A stakeholder analysis with social network analysis to develop a community pharmacy service, 2019]

A Snapshot Of The Difficulties Inherent In Requirements Elicitation



STAKEHOLDER

- Conflicting needs
- Fear of the unknown
- Changing priorities

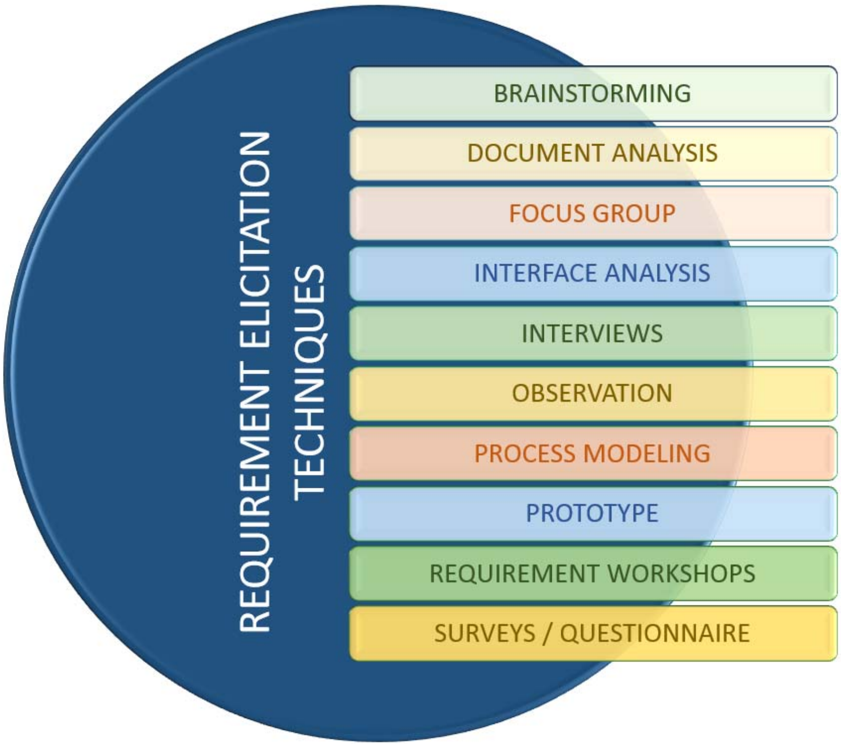
+ Assumptions
+ Different concerns
+ Limited experience
+ Project complexity
+ Articulation problems
+ Different vocabularies
+ Omission of the obvious
+ Ambiguity of requirements
+ Limited domain knowledge
+ Limited technical knowledge

ANALYST

- Bounded rationality
- Lack of empathy
- Poor listening skills

[A beginner's guide to requirements elicitation, 2015]

Some requirement elicitation techniques



[www.anarsolutions.com]

Requirements… how to see those

In such cases negotiations are needed…

ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.3431 requirement

1. statement that translates or expresses **a need and its associated constraints and conditions** *[ISO/IEC TS 24748-1:2016 Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management,* 2.41; *ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering,* 4.1.19*]*

2. **condition or capability that must be met or possessed** by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents *[IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes,* 3.2*]*

3. **provision that contains criteria** to be fulfilled *[ISO/IEC 14143-2:2011 Information technology — Software measurement — Functional size measurement — Part 2: Conformity evaluation of software size measurement methods to ISO/IEC 14143-1,* 3.10*]*

4. **a condition or capability that must be present in a product**, service, or result to satisfy a contract or other formally imposed specification *[A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fifth Edition] cf.* design requirement, functional requirement, implementation requirement, interface requirement, performance requirement, physical requirement.

EXAMPLE: software component requirement

ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.3434

requirements analysis

1. process of **studying user needs** to arrive at a definition of system, hardware, or software requirements

2. process of studying and refining system, hardware, or software requirements

3. systematic investigation of user requirements to arrive at a definition of a system *[ISO/IEC 2382:2015, Information technology — Vocabulary]*

4. determination of product- or service-specific performance and functional characteristics based on analyses of customer needs, expectations, and constraints; operational concept; projected utilization environments for people, products, services, and processes; and measures of effectiveness.

ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.3439 requirements elicitation

1. process through which the acquirer and the suppliers of a system **discover, review, articulate, understand, and document the requirements on the system and the life cycle processes** *[ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering,* 4.1.18*]*

2. use of systematic techniques, such as prototypes and structured surveys, to proactively identify and document customer and end-user needs.

3.3447 requirements specification

1. **document** that specifies the requirements for a system or component *cf.* design description, functional specification, performance specification

Note 1 to entry: Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards.

ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary

3.4089

SyRS

1. System Requirement Specification *[ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering,* 4.2*]*

cf. SRS.

Requirements traceability is needed only in ”mission-critical” systems, e.g. satellite systems and healthcare machines. There are special software for requirements management.

ISO / IEC / IEEE 24765:2017

3.1581

feasibility study

1. study to **identify and analyze a problem and its potential solutions** in order to determine their viability, costs, and benefits.

3.3431

requirement

1. statement that translates or expresses a need and its associated constraints and conditions

2. condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents

3. provision that contains criteria to be fulfilled

4. a condition or capability that must be present in a product, service, or result to satisfy a contract or other formally imposed specification.

Tampereen yliopisto
Tampere University

ISO 24765:2017 Systems and software engineering — Vocabulary

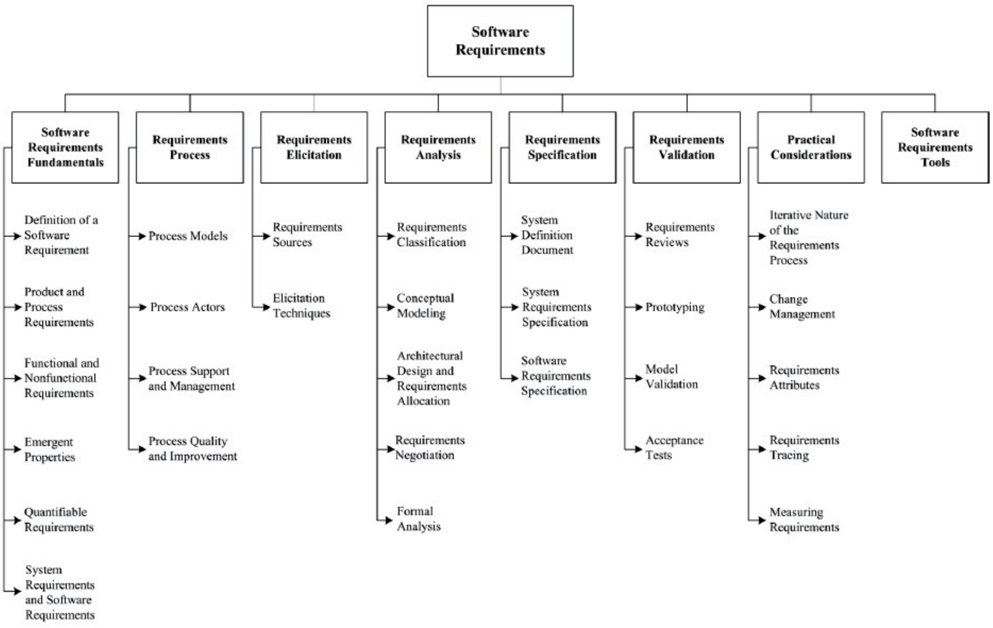
3.1582 **feature** (FI: ominaisuus)

- ***1.*** distinguishing characteristic of a system item
- ***2.*** functional or non-functional distinguishing characteristic of a system, often an enhancement to an existing system
- ***3.*** abstract functional characteristic of a system of interest that end-users and other stakeholders can understand

3.1716 **functionality** (FI: toiminnallisuus)

- ***1.*** capabilities of the various computational, user interface, input, output, data management, and other features provided by a product
- Note 1 to entry: This characteristic is concerned with what the software does to fulfill needs. The software quality characteristic functionality can be used to specify or evaluate the suitability, accuracy, interoperability, security, and compliance of a function.

1-2 *SWEBOK® Guide* V3.0

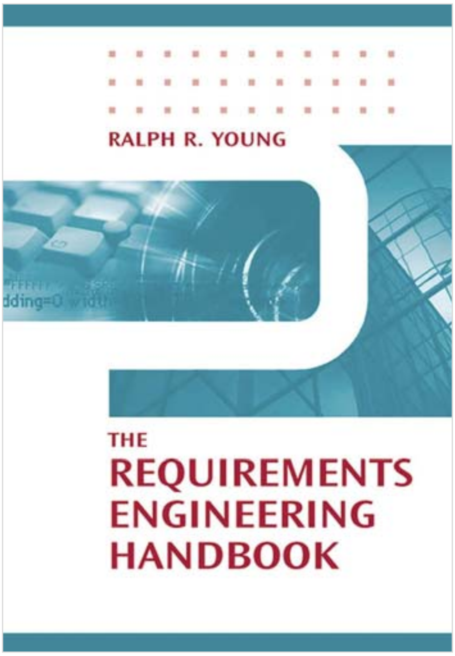


- **Software Requirements**
  - **Software Requirements Fundamentals**
    - Definition of a Software Requirement
    - Product and Process Requirements
    - Functional and Nonfunctional Requirements
    - Emergent Properties
    - Quantifiable Requirements
    - System Requirements and Software Requirements
  - **Requirements Process**
    - Process Models
    - Process Actors
    - Process Support and Management
    - Process Quality and Improvement
  - **Requirements Elicitation**
    - Requirements Sources
    - Elicitation Techniques
  - **Requirements Analysis**
    - Requirements Classification
    - Conceptual Modeling
    - Architectural Design and Requirements Allocation
    - Requirements Negotiation
    - Formal Analysis
  - **Requirements Specification**
    - System Definition Document
    - System Requirements Specification
    - Software Requirements Specification
  - **Requirements Validation**
    - Requirements Reviews
    - Prototyping
    - Model Validation
    - Acceptance Tests
  - **Practical Considerations**
    - Iterative Nature of the Requirements Process
    - Change Management
    - Requirements Attributes
    - Requirements Tracing
    - Measuring Requirements
  - **Software Requirements Tools**

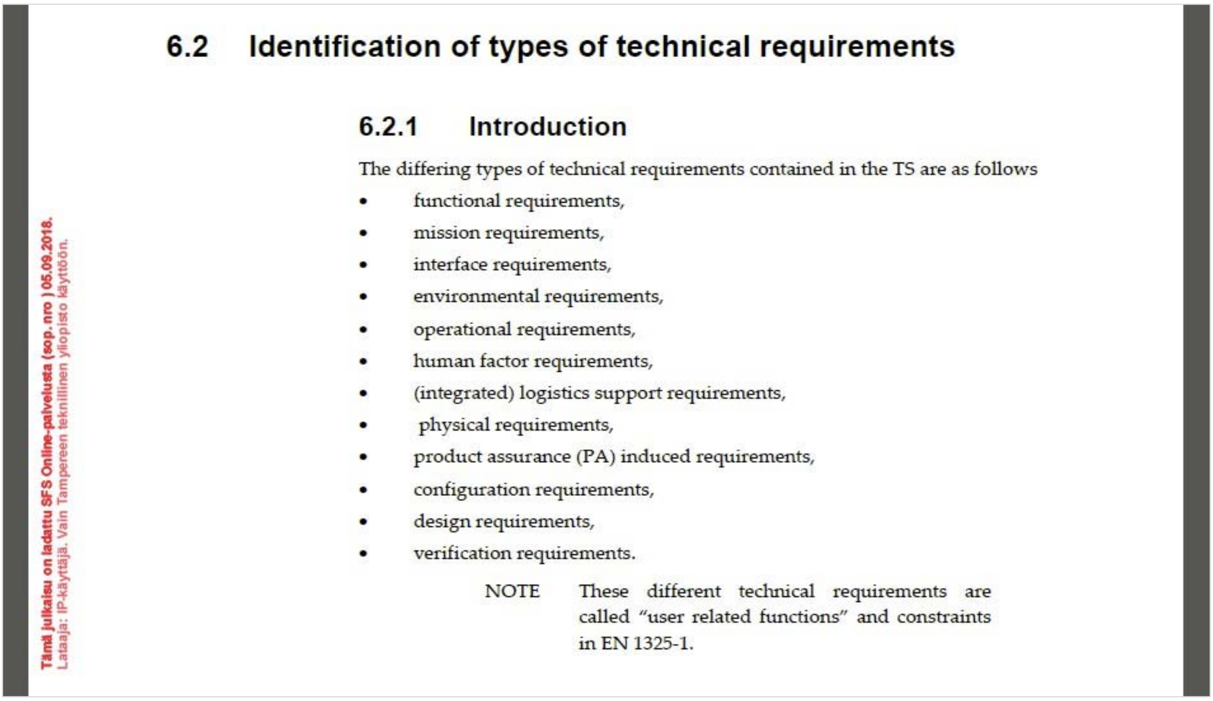
Figure 1.1. Breakdown of Topics for the Software Requirements KA

Definitions and Descriptions of Requirements Types

- *Business Requirements*
- *Stated Requirements Versus Real Requirements*
- *User Requirements*
- *High-Level or System-Level Requirements*
- *Business Rules*
- *Functional Requirements*
- *Nonfunctional Requirements*
- *Derived Requirements*
- *Design Requirements and Design Constraints*
- *Performance Requirements*
- *Interface Requirements*
- *Verified Requirements*
- *Validated Requirements*
- *Qualification Requirements*
- *The "Ilities" and Specialty Engineering Requirements*
- *Unknowable Requirements*
- *Product Requirements*
- *Process Requirements*
- *Logistics Support Requirements*
- *Environmental Requirements*
- *System, Subsystem, and Component Requirements.*



SFS-EN 16603-10-06:2014



6.2 Identification of types of technical requirements

6.2.1 Introduction

The differing types of technical requirements contained in the TS are as follows

- functional requirements,
- mission requirements,
- interface requirements,
- environmental requirements,
- operational requirements,
- human factor requirements,
- (integrated) logistics support requirements,
- physical requirements,
- product assurance (PA) induced requirements,
- configuration requirements,
- design requirements,
- verification requirements.

NOTE These different technical requirements are called "user related functions" and constraints in EN 1325-1.

requirements, three main cases

Functional requirements , e.g.

- user's functionalities
- inputs and outputs
- browser Front and Back buttons are disabled

Non-functional requirements , e.g.

- GUI is as customer's guidebook XYZ-2019-A, coding conventions
- localisation, internationalisation = I(18)N
- safety and security

Restrictions and limitations , e.g.

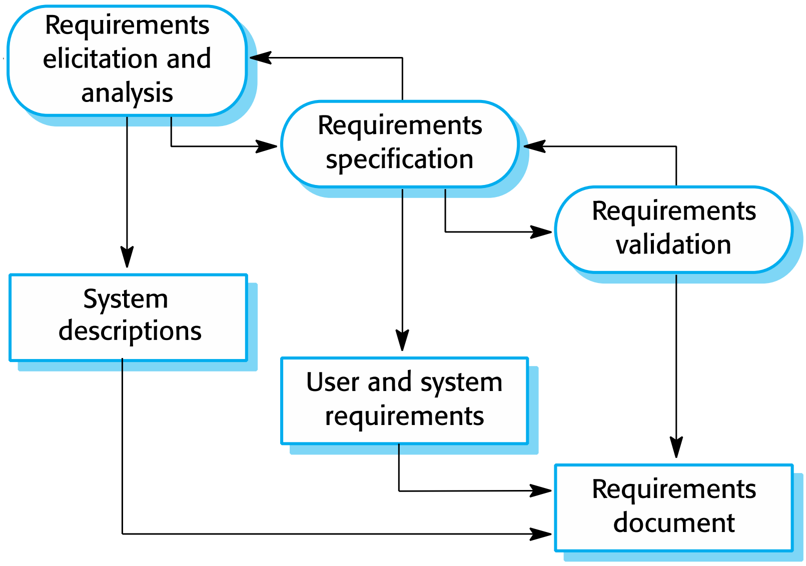
- compatible with customer's old system (DB)
- supported operating systems and browsers
- project schedule and budget.

In some cases a requirement may be considered as non-functional or restriction.

The requirements engineering process

[Sommerville 2014]

Requirements engineering is developing a software specification.


Requirements elicitation and analysis
Requirements specification
Requirements validation
System descriptions
User and system requirements
Requirements document

Functional and non-functional requirements

[Sommerville 2014]

✧ Functional requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- May state what the system should not do.

✧ Non-functional requirements

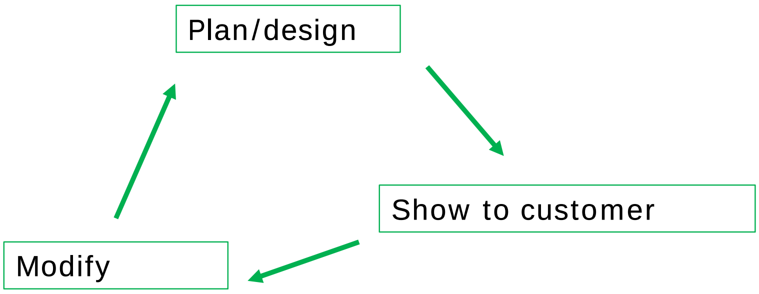
- Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Often apply to the system as a whole rather than individual features or services.

✧ Domain requirements

- Constraints on the system from the domain of operation.

start quickly, small steps, iterate

- When dealing with customer who doesn’t actually know exactly what (s)he wants, make demos/prototypes quickly and often.
- When showing first hand-written sketches of GUI to customer after two weeks, you always get some comments (yes/no). Then update your demo/requirements and show next version after 1..2 weeks.
- It is better to show your ideas of the software system often to customer, instead of thinking yourself many weeks and fine-tuning a demo you THINK customer wants.
- Work in iterations.

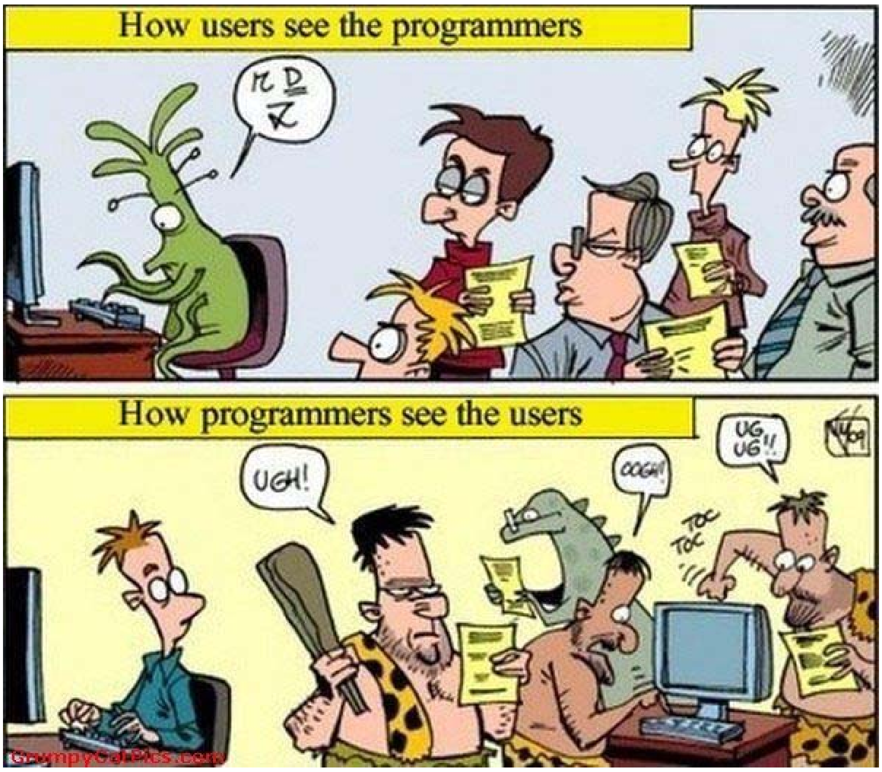

Plan/design

Show to customer

Modify

By the way, remember WHY you do the software, and for WHOM.

”User before technology…?”

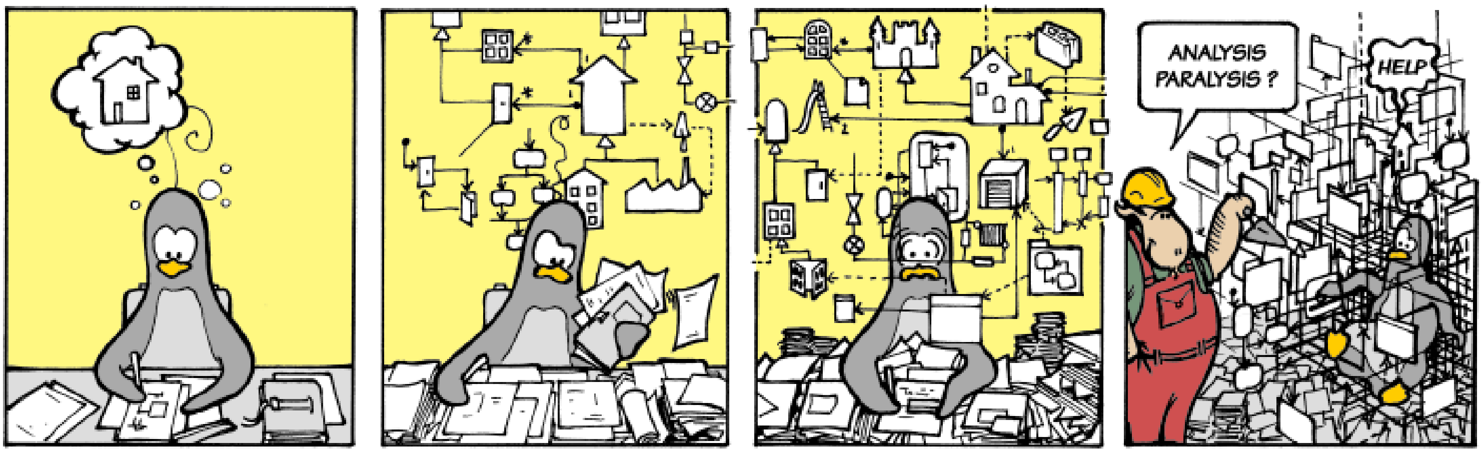


Which car radio would user prefer… ?
Which car radio would developer prefer… ?

10 functions, 100 €

30 functions, 300 €

Agile methods bring help to ”analysis paralysis”



Feature creep = more and more features are added,
but… project budget and schedule are not updated accordingly.

Kahoot query (quiz);

What is a good DI (Master) like ?

- every graduated DI is good !
- the DI who has grade 4 or 5
- the DI who graduated within five years
- the DI who has full-time work at graduation.

FI: DI = diplomi-insinööri (Master of Science, tech/eng).

Just do what is required, don’t mess

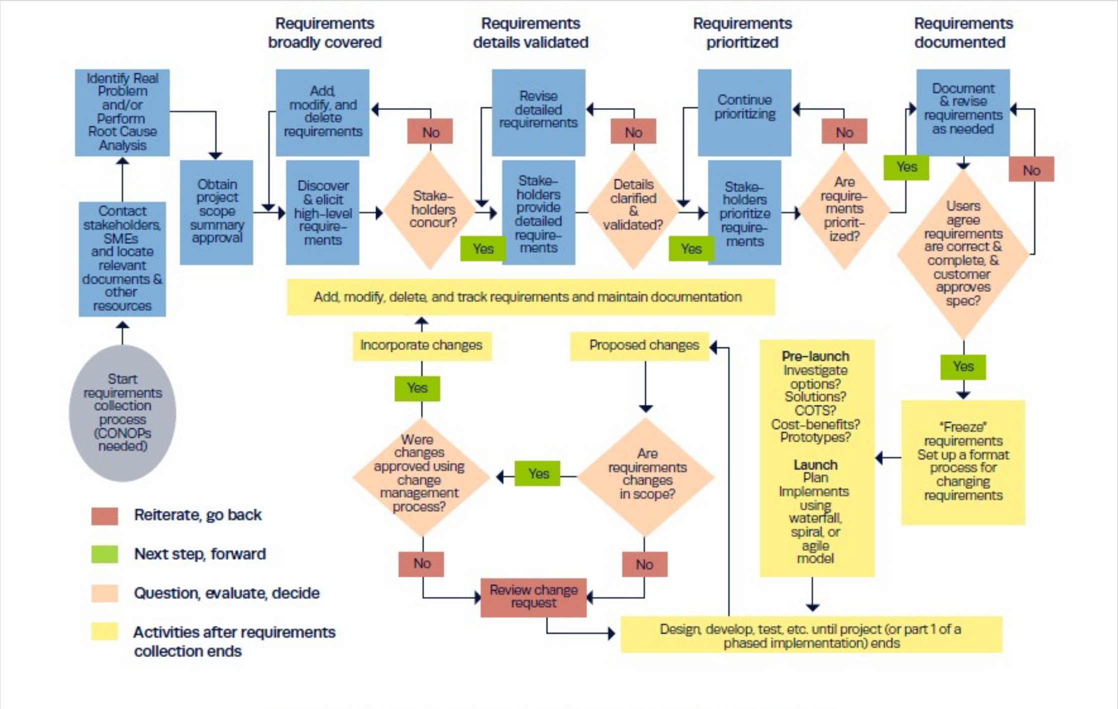
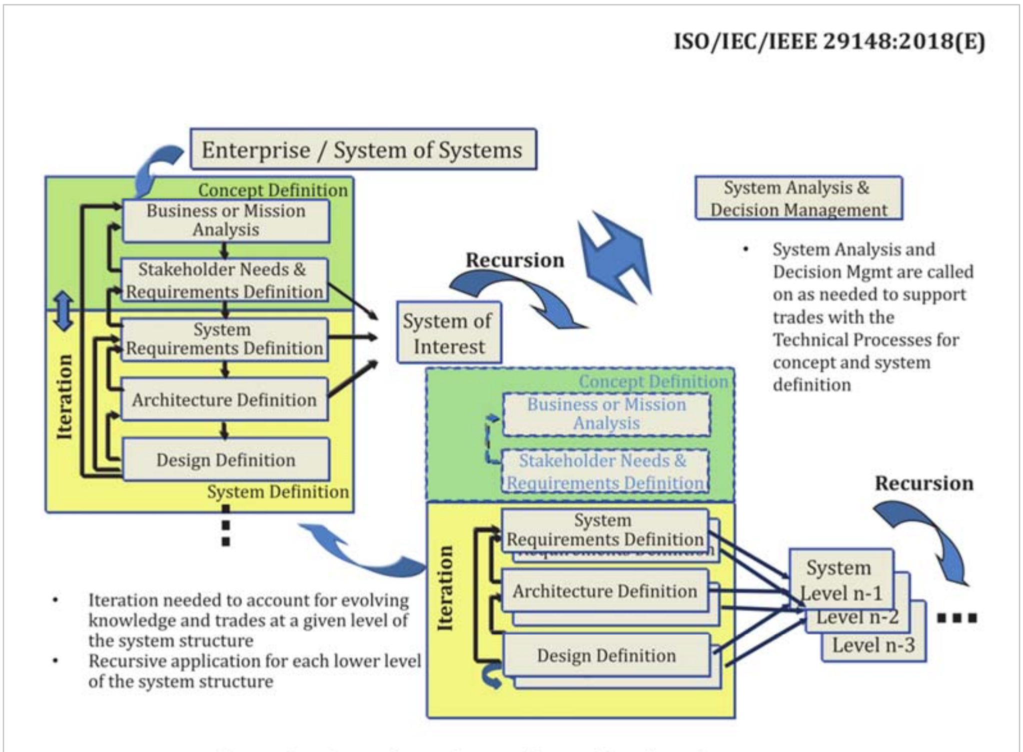

Requirements broadly covered
Requirements details validated
Requirements prioritized
Requirements documented
Identify Real Problem and/or Perform Root Cause Analysis
Contact stakeholders, SMEs and locate relevant documents & other resources
Start requirements collection process (CONOPs needed)
Obtain project scope summary approval
Add, modify, and delete requirements
Discover & elicit high-level require-ments
Stake-holders concur?
No
Yes
Revise detailed requirements
Stake-holders provide detailed require-ments
Details clarified & validated?
No
Yes
Continue prioritizing
Stake-holders prioritize require-ments
Are require-ments priorit-ized?
No
Yes
Document & revise requirements as needed
Users agree requirements are correct & complete, & customer approves spec?
No
Yes
Add, modify, delete, and track requirements and maintain documentation
Incorporate changes
Yes
Were changes approved using change management process?
No
Proposed changes
Are requirements changes in scope?
Yes
No
Review change request
Pre-launch
Investigate options? Solutions? COTS? Cost-benefits? Prototypes?
Launch
Plan Implements using waterfall, spiral, or agile model
"Freeze" requirements Set up a format process for changing requirements
Design, develop, test, etc. until project (or part 1 of a phased implementation) ends
🟥 Reiterate, go back
🟩 Next step, forward
🟧 Question, evaluate, decide
🟨 Activities after requirements collection ends

Figure 1. Overview of Requirements Collection and Change Processes

CONOPS = concept of operations, document. [MITRE, 2014]

ISO 29148:2018
Systems and software engineering —
Life cycle processes
— Requirements engineering


ISO/IEC/IEEE 29148:2018(E)
Enterprise / System of Systems
Concept Definition
Business or Mission Analysis
Stakeholder Needs & Requirements Definition
System Requirements Definition
Architecture Definition
Design Definition
System Definition
Iteration
Recursion
System of Interest
System Analysis & Decision Management
Concept Definition
Business or Mission Analysis
Stakeholder Needs & Requirements Definition
System Requirements Definition
Architecture Definition
Design Definition
Iteration
Recursion
System Level n-1
Level n-2
Level n-3

- System Analysis and Decision Mgmt are called on as needed to support trades with the Technical Processes for concept and system definition

- Iteration needed to account for evolving knowledge and trades at a given level of the system structure
- Recursive application for each lower level of the system structure

Figure 2 — Iterative and recursive application of processes

ISO/IEC/IEEE 29148:2018(E)

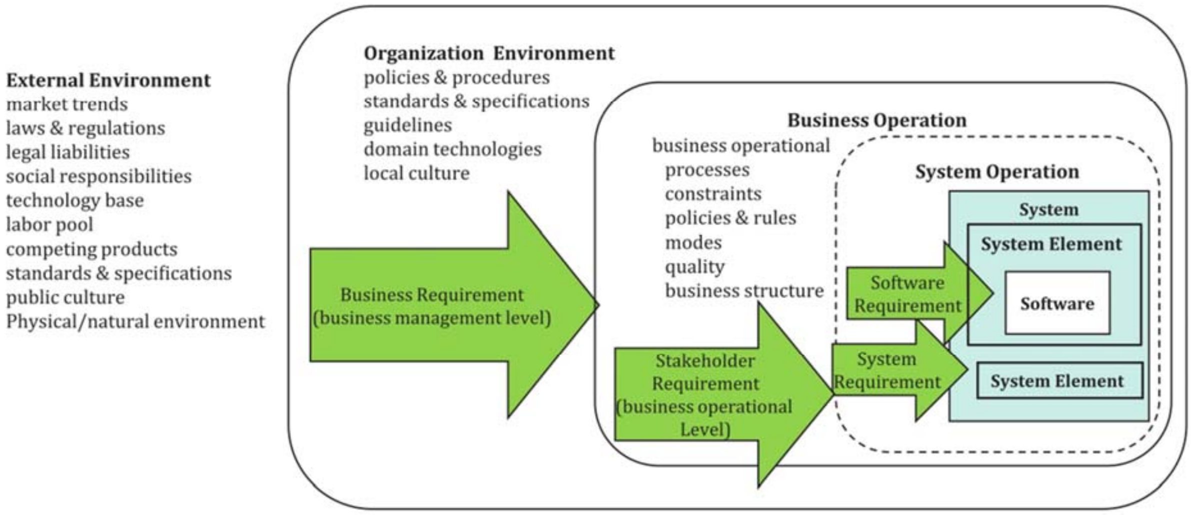


Figure 3 — Example of requirements scope in a business context

ISO/IEC/IEEE 29148:2018(E)

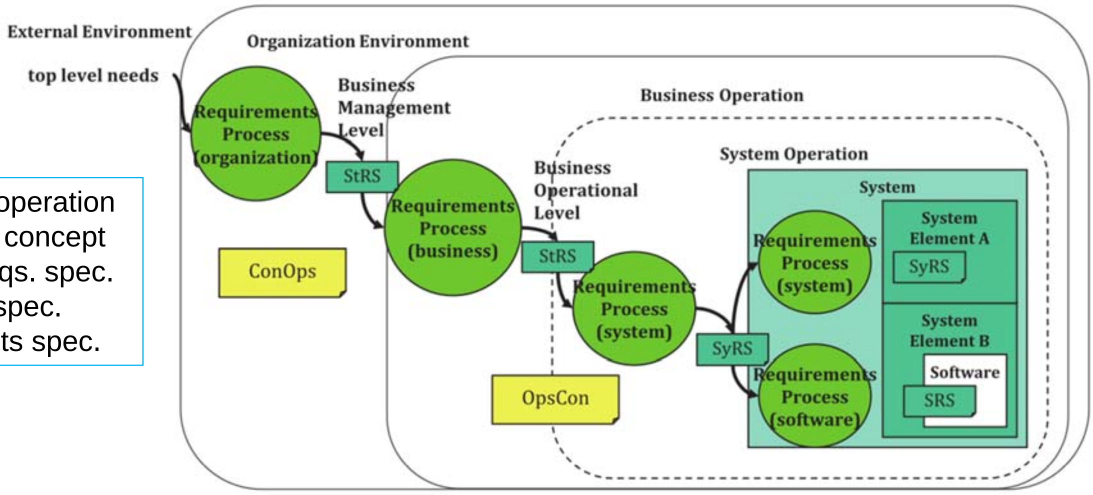
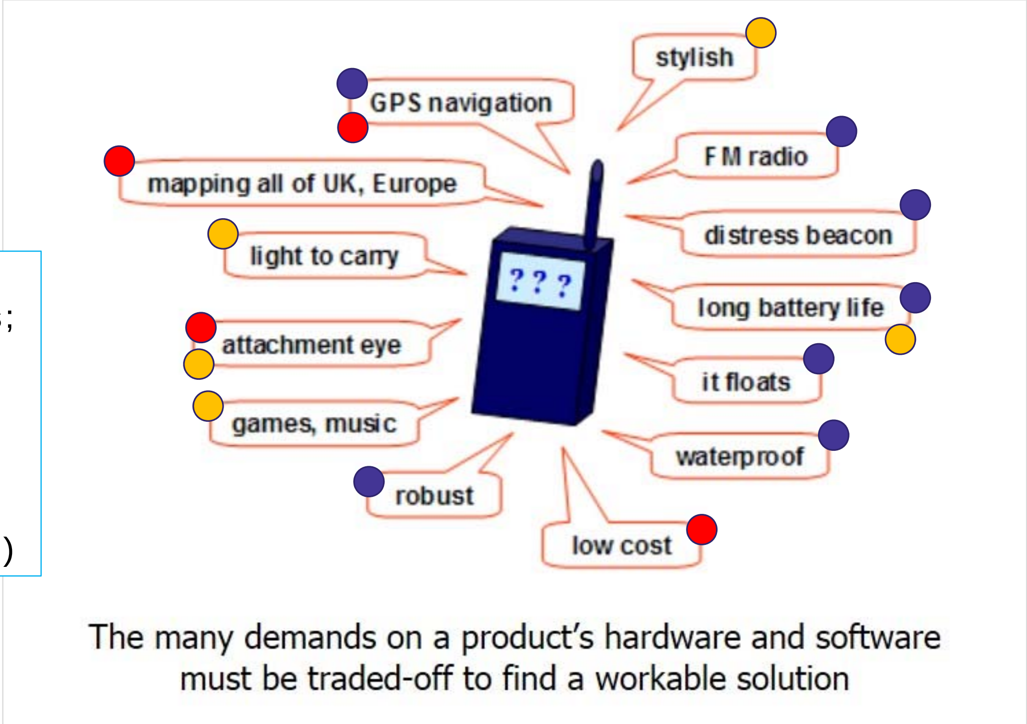


Figure 4 — Example of the relationships between requirements processes and specifications

ConOps = concept of operation
OpsCon = operational concept
StRS = stakeholder reqs. spec.
SyRS = system reqs. spec.
SRS = Sw requirements spec.

Contradicting requirements and goals;
- 🔵 technical goals
- 🔴 business goals
- 🟡 user’s goals
- etc.

(this is just an example)



The many demands on a product’s hardware and software must be traded-off to find a workable solution

Agile methods help in this
- start with basic features
- develop and test in iterations
- show demo/proto to customer
- customer gives feedback
- modify demo/proto for next version.

ANALYSIS PARALYSIS

The state of over-analysing (or over-thinking) a situation so that a decision or action is never taken, in effect paralysing the outcome.

STEPS TO OVERCOME

HAVE A CLEAR DEFINED GOAL
GET THE BASICS RIGHT
abc
SIMPLIFY YOUR WORKOUTS
MAKE A START! PERFECT IS THE ENEMY OF GOOD
TAKE SMALL STEPS
SET LIMITS AND HOLD YOURSELF ACCOUNTABLE
@BODYSMARTFITNESS

Christopher Lindquist: **Software Requirements Mess**

Hugh Cumming had his work cut out for him. The gap between what his not-yet-implemented call center management application at a large European company could do and the requirements list created by 40 eager business-side stakeholders now filled 3,000 pages and threatened to delay an already overdue call center consolidation effort another four to five years. "My first instinct was that the project had absolutely no chance of success," says Cumming, currently CIO for ADP Employer Services Canada.

Requirements, as every CIO knows, are a problem, but CIOs may not be aware of just how catastrophic the problem has become. **Analysts report that as many as 71 percent of software projects that fail do so because of poor requirements management**, making it the single biggest reason for project failure—bigger than bad technology, missed deadlines or change management fiascoes. Though CIOs are rarely directly responsible for requirements management, they are accountable for poor outcomes, which, when requirements go bad, can include: project delays, software that doesn't do what it's supposed to and, worst of all, software that may not work correctly when rolled out, putting the business—and the CIO's job—at risk.

Mishandled requirements can torpedo a project at any time, from inception to delivery. **Start down the wrong road and you arrive at the wrong destination. And even if you're heading in the right direction, making fumbling changes midstream can be almost as deadly.**

CIO = Chief Information Officer (FI: tietohallintojohtaja)

Chris Doig: Five ways inadequate requirements wreak havoc with enterprise software purchases

1) Inadequate functionality

2) Discovering “new” requirements

3) Implementing “new” requirements

4) Business disruption

5) Unmet expectations.

Requirements will be discovered during the initial requirements analysis, during the implementation or in early production use. An implementation project that starts with a comprehensive list of ALL significant requirements, who wants them, why they are wanted and how important they are, enables the project manager to create a realistic implementation plan. When no significant new requirements are discovered, the implementation is on schedule and within budget.

[www.cio.com]

Requirements may / will / surely change…

Good specification is

- complete
- accurate, specific
- faultless, soundness
- understandable
- testable (measurable, to fulfill requirement)
- traceable
- no redundancy
- as short as possible.

If there would be mind-reading, requirements documentation would be obsolate.

Requirements (TS = technical specification) should be unambiguous/unequivocal

8.3.3 Format restrictions

a. List of terms that shall not be used in a TS requirement (technical requirements specification) “and/or”, “etc.”, “goal”, “shall be included but not limited to”, “relevant”, “necessary”, “appropriate”, “as far as possible”, “optimize”, “minimize”, “maximize”, “typical”, “rapid”, “userfriendly”, “easy”, “sufficient”, “enough”, “suitable”, “satisfactory”, “adequate”, “quick”, “first rate”, “best possible”, “great”, “small”, “large”, and “state of the art”.

[SFS-EN 16603-10-06:2014, SPACE ENGINEERING. PART 10-06: TECHNICAL REQUIREMENTS SPECIFICATION]

What to write about a requirement

- date (of origin)
- author/writer
- source (customer and why…)
- type (change, addition, correction)
- description
- dependence to/from other requirements (if any)
- need (required, optional = nice to have, extra = additional)
- propability of change (will not, maybe, is likely) during requirements process.

If requirements change, project plan should cange, too

Government design principles

The UK government's design principles and examples of how they've been used.
Published 3 April 2012
From: Government Digital Service

Contents

1. Start with user needs
2. Do less
3. Design with data
4. Do the hard work to make it simple
5. Iterate. Then iterate again
6. This is for everyone
7. Understand context
8. Build digital services, not websites
9. Be consistent, not uniform
10. Make things open: it makes things better.

https://www.gov.uk/guidance/government-design-principles

a few more requirement’s attributes

Safety = e.g. safe to use, for users (sw does nothing it should not do, warns user).

Security = e.g. information security, cyber security (architectural issues, user rights).

Usability = e.g. easy to use, user understands how to do some task.

Developer’s and customer’s responsibilities

Developer’s responsibilities are e.g.

- take all aspects into consideration; user groups, stakeholders,...
- make demos/prototypes often to customer
- in case of major changes, request updates to project plan (schedule, budget,...).

Customer’s responsibilities are e.g.

- be committed, not just involved
- comment requirements and GUI to developer; just ”OK” is not a good answer
- be loyal to fellow workres/users, not just want ”personal” features to sw
- in case of major changes, request updates to project plan (schedule, budget,...).

requirements management

3.3442

requirements management

1. activities that ensure **requirements are identified, documented, maintained, communicated and traced** throughout the life cycle of a system, product, or service *[ISO/IEC/IEEE 29148:2011 Systems and software engineering — Life cycle processes — Requirements engineering,* 4.1.20*]*

2. provision of storing and editing capabilities, tracking history of edition, versioning, author identification, change management, time stamping, user notification for content changes, security rights control *[ISO/IEC TR 24766:2009 Information technology — Systems and software engineering — Guide for requirements engineering tool capabilities,* 3.3*] cf.* software requirements management

[ISO/IEC/IEEE 24765:2017]

requirements traceability

3.3449 requirements traceability

1. identification and documentation of the derivation path (upward) and allocation/ flow-down path (downward) of requirements in the requirements hierarchy

2. discernible association between a requirement and related requirements, implementations, and verifications

3. traceabilities in domain and application requirements respectively and those between them *[ISO/IEC 26551:2016 Software and systems engineering — Tools and methods for product line requirements engineering,* 3.18*]*

3.3450 requirements traceability matrix (RTM)

1. table that links requirements to their origin and traces them throughout the project life cycle

2. a grid that links product requirements from their origin to the deliverables that satisfy them *[A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fifth Edition]*

[ISO/IEC/IEEE 24765:2017]

Further reading…

Remember ”Additional material” at Moodle

(no exam questions about ”Additional material”)

- there is material also about requirements

Highlights - What to remember

- requirements (= **WHAT** the system is supposed to do) are crucial
- if there would be mind-reading, we wouldn’t need requirements specification
- end-users might know best what they need
- several stakeholders may have different goals and needs
- requirements may be, and usually are somehow, **contradicting** (trade-offs)
- beware of analysis paralysis – do it agile way, start now and show proto to customer
- during a project, requirements (customer’s mind or external interactions) do change
- also outside world (e.g. business) may change during project’s time
- customer’s **commitment** is needed !